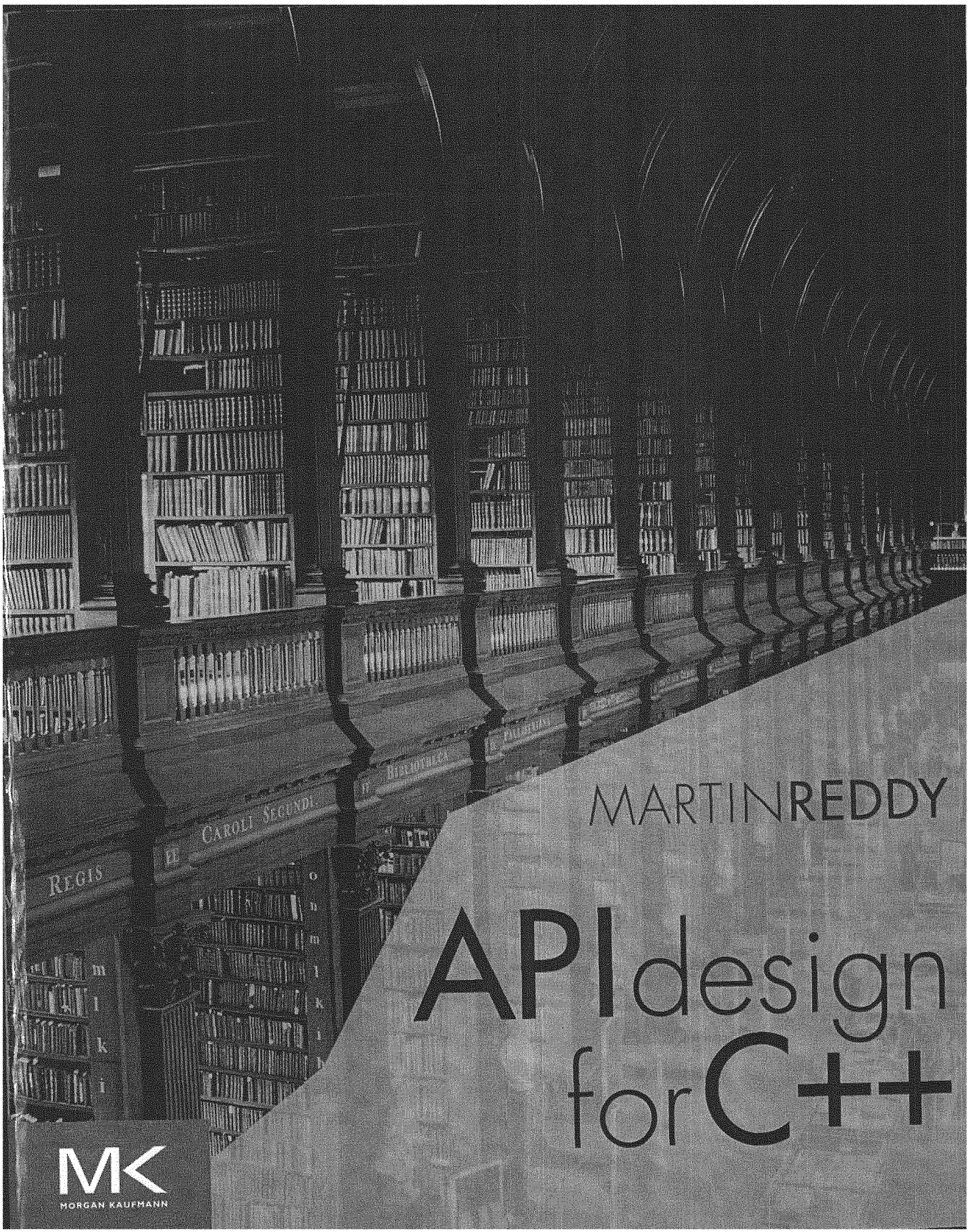
MARTIN REDDY
API design for C++
MORGAN KAUFMANN

Acquiring Editor: Todd Green
Editorial Assistant: Robyn Day
Project Manager: André Cuello
Designer: Eric DeCicco

Morgan Kaufmann is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods or professional practices may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information or methods described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
Application submitted

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 978-0-12-385003-4

Printed in the United States of America
11 12 13 14 10 9 8 7 6 5 4 3 2 1

For information on all MK publications visit our website at www.mkp.com

As an example of a well-known API, Microsoft's Windows API (often referred to as the Win32 API) is a collection of C functions, data types, and constants that enable programmers to write applications that run on the Windows platform. This includes functions for file handling, process and thread management, creating graphical user interfaces, talking to networks, and so on.

The Win32 API is an example of plain C API rather than a C++ API. While you can use a C API directly from a C++ program, a good example of a specific C++ API is the Standard Template Library (STL). The STL contains a set of container classes, iterators for navigating over the elements in those containers, and various algorithms that act on those containers (Josuttis, 1999). For instance, the collection of algorithms includes high-level operations such as `std::search()`, `std::reverse()`, `std::sort()`, and `std::set_intersection()`. The STL therefore presents a logical interface to the task of manipulating collections of elements, without exposing any of the internal details for how each algorithm is implemented.

TIP

An API is a logical interface to a software component that hides the internal details required to implement it.

1.2 WHAT'S DIFFERENT ABOUT API DESIGN?

Interfaces are the most important code that a developer writes. That's because problems in an interface are far more costly to fix than problems in the associated implementation code. As a result, the process of developing shared APIs demands more attention than standard application or Graphical User Interface (GUI) development. Of course, both should involve best design practices; however, in the case of API development, these are absolutely critical to its success. Specifically, some of the key differentiating factors of API development include the following.

- An API is an interface designed for developers, in much the same way that a GUI is an interface designed for end users. In fact, it's been said that an API is a user interface for programmers (Arnold, 2005). As such, your API could be used by thousands of developers around the world, and it will undoubtedly be used in ways that you never intended (Tulach, 2008). You must anticipate this in your design. A well-designed API can be your organization's biggest asset. Conversely, a poor API can create a support nightmare and even turn your users toward your competitors (Bloch, 2005), just as a buggy or difficult-to-use GUI may force an end user to switch to a different application.
- Multiple applications can share the same API. Figure 1.1 showed that a single application can be composed of multiple APIs. However, any one of those APIs could also be reused in several other applications. This means that while problems in the code for any given application will only affect that one application, errors in an API can affect all of the applications that depend on that functionality.
- You must strive for backward compatibility whenever you change an API. If you make an incompatible change to your interface, your clients' code may fail to compile, or worse their code could compile but behave differently or crash intermittently. Imagine the confusion and chaos that would arise if the signature of the `printf()` function in the standard C library was